Deep Learning Introduction

Christian Szegedy
Geoffrey Irving

Google Research

Machine Learning

Supervised Learning Task

- Assume $G : D \longrightarrow P$
- $f : D \times M \longrightarrow P$
- $\sigma : P \times P \longrightarrow \mathbb{R}$
- $S \subset D \times P$

Ground truth G
Model architecture f
Prediction metric σ
Training samples S

Find model parameters $m \epsilon M$ such that the expected

$\mathbb{E}(\sigma(F(d, m), G(d))$ is minimized

Machine Learning

Unsupervised Learning

Set of tasks that work on the uncurated data.

Predict properties that are inherently present in the data alone.

Machine Learning

Generative Learning Task

- $\mu : \Omega(D) \longrightarrow [0, 1]$ Input space with probability measure
- $f : [0, 1]^n \times M \longrightarrow D$ Generative model architecture

Find model parameters $m \in M$ such that: $\mu\,(f(S, m)) \sim \mu'\,(S)$

Machine Learning

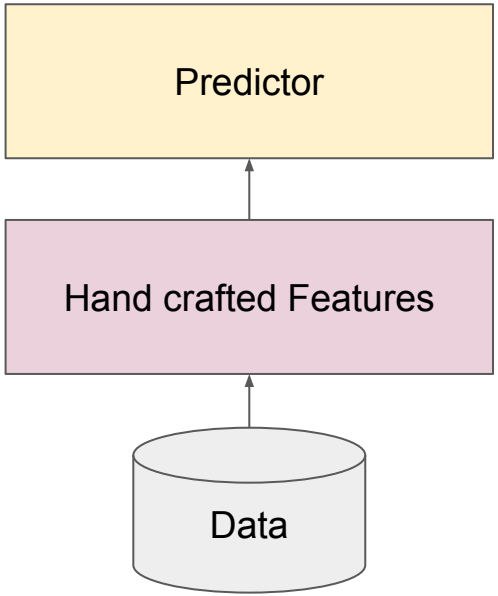
Supervised Learning as Marginal Computation

- $\mu : \Omega(D \times P) \longrightarrow [0, 1]$
- $f : [0, 1]^n \times D \times M \longrightarrow P$

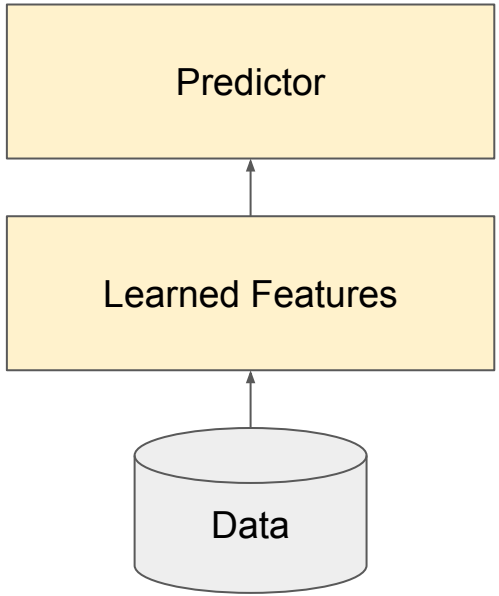
Expanded Input space
Conditional generative model.

Find model parameters $m \in M$ such that: $\mu\,(f(S, d, m)|d) \sim \mu'\,(S)$

Deep versus Shallow Learning

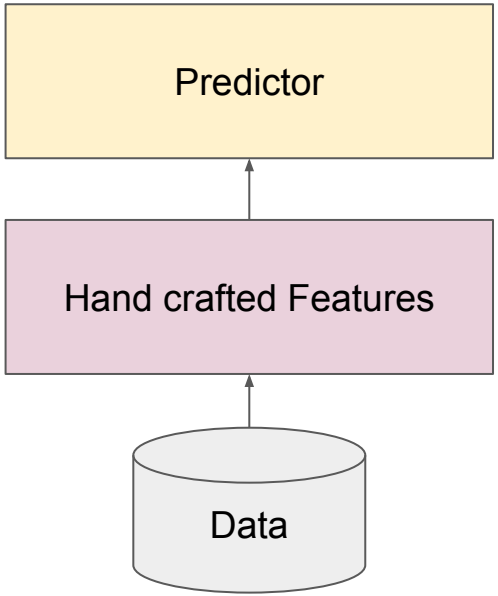


Traditional machine learning

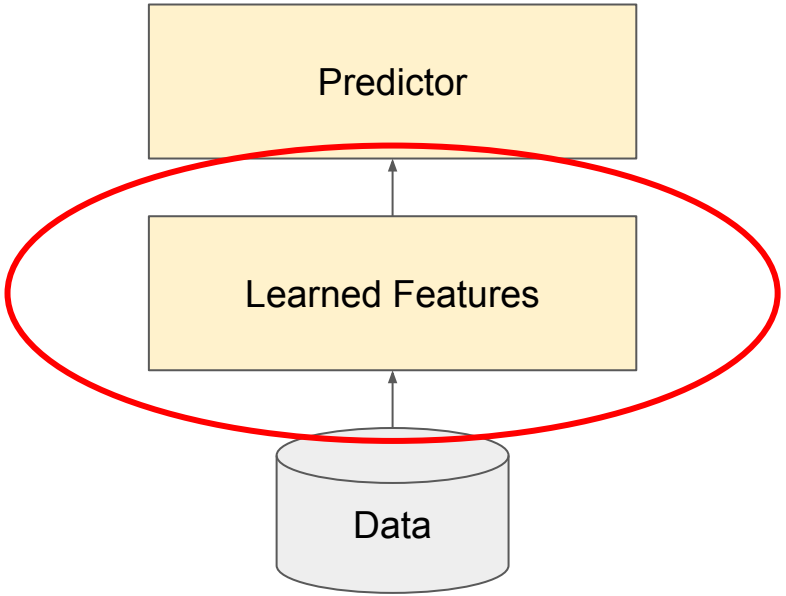


Deep Learning

Deep versus Shallow Learning

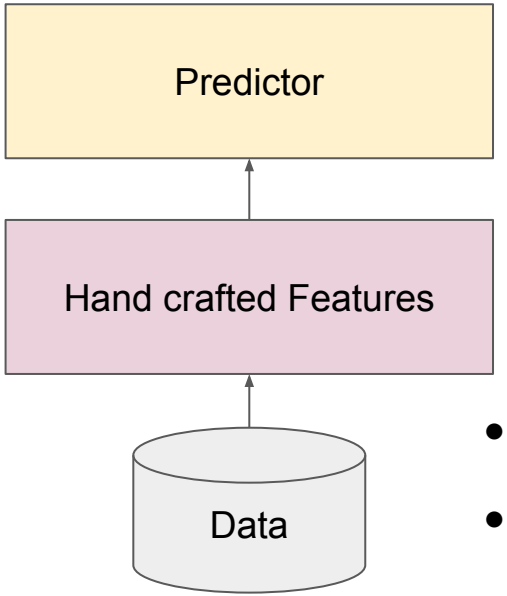


Traditional machine learning



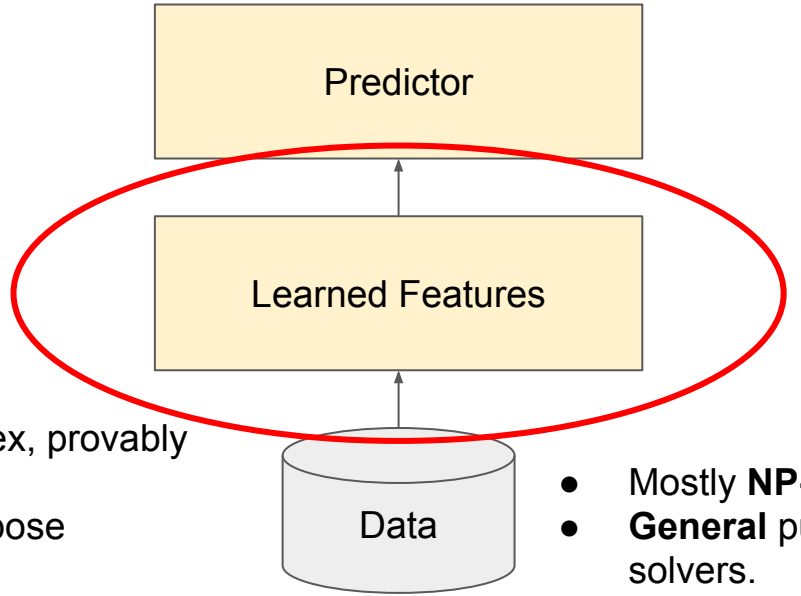
Deep Learning

Deep versus Shallow Learning



Traditional machine learning

- Mostly convex, provably **tractable**.
- **Special** purpose solvers.
- **Non-layered** architectures.



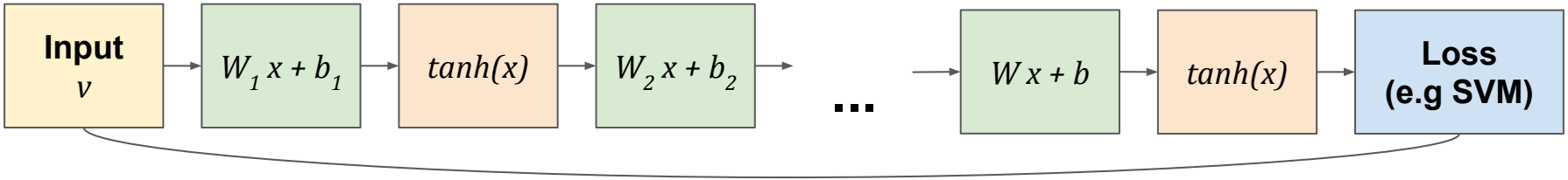
Deep Learning

- Mostly **NP-Hard**
- **General** purpose solvers.
- **Hierarchical** models

Provably Tractable Deep Learning Approaches

- **Sum-Product networks** [by Hoifung Poon and Pedro Domingos]
  - Can learn generative models
  - Hierarchical structure
  - Automated learning of low level features
  - Tractable training/inference under certain conditions
  - Practical implementations
- **Provable Bounds for Learning Some Deep Representations** [Sanjeev Arora, Aditya Bhaskara, Rong Ge and Tengyu Ma]
  - Can learn generative models.
  - Hierarchical structure
  - Automated learning of low level features
  - Provably tractable for extremely sparse graphs
  - Creates deep and sparse artificial neural networks
  - Based on the polynomial time solvable graph-square-root problem.

Classical Feed-Forward Artificial Neural Networks



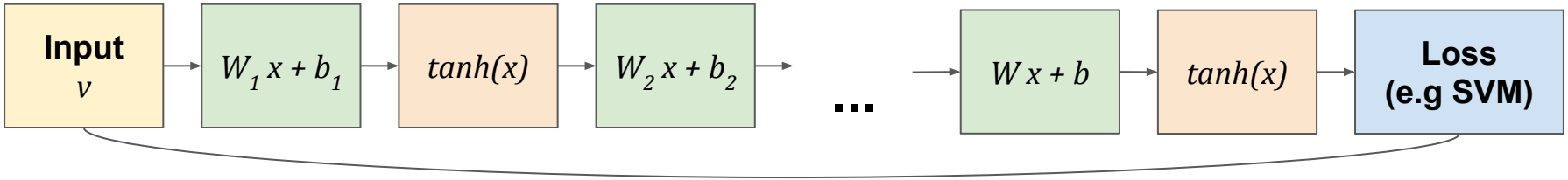
Each sample is a vector

(Element-wise nonlinearity)

Minimize $\sum_v loss(N(W_1, b_1, \ldots, W_n, b_n; v))$

Multilayer **perceptron** [Frank Rosenblatt, 1961]

Classical Feed-Forward Artificial Neural Networks



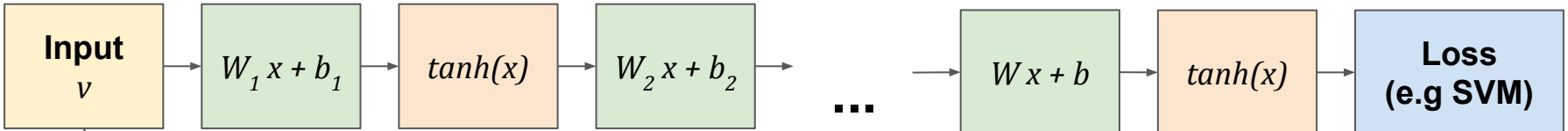
Each sample is a vector

(Element-wise nonlinearity)

Minimize $\sum_v loss(N(W_1, b_1, \ldots, W_n, b_n; v))$

In today's networks, tanh is increasingly replaced by max(x, 0)
(**Rectified linear units** or ReLUs)

Classical Feed-Forward Artificial Neural Networks



Each sample is a vector

(Element-wise nonlinearity)

Minimize $\sum_v loss(N(W_1, b_1, \ldots, W_n, b_n; v))$

A highly nonlinear function!

Huge Sum, ranges over all training examples!

Optimizing the Neural Network Parameters

With $M = (W_1, b_1, \ldots, W_n, b_n)$

Minimize $\sum_v loss(N(M, v))$

Optimizing the Neural Network Parameters

With $M = (W_1, b_1, \ldots, W_n, b_n)$

Minimize $\sum_v loss(N(M, v))$

Use gradient descent in the parameter space:

$$M_{i+1} \longleftarrow M_i - \alpha \frac{\partial}{\partial M} \sum_v loss(N(M_i, v))$$

Stochastic Gradient Descent

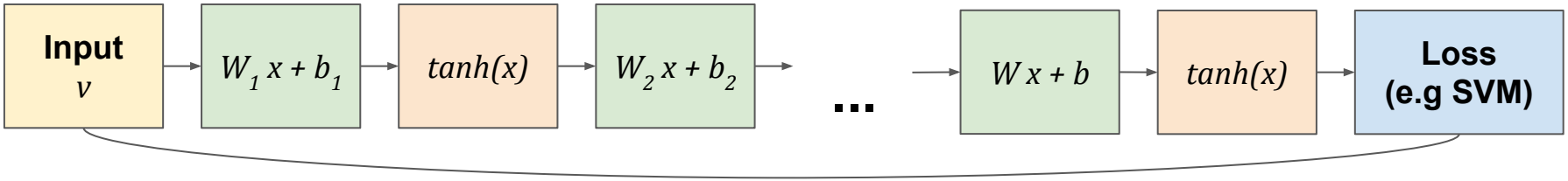
$$M_{i+1} \longleftarrow M_i - \alpha \frac{\partial}{\partial M} \sum_{v} loss(N(M_i, v))$$

Learning rate α

$$M_{i+1} \longleftarrow M_i - \alpha \frac{\partial}{\partial M} \sum_{v \in B_i} loss(N(M_i, v))$$

Randomly sampled **Minibatch** B_i

Compute derivatives via chain rule



Each sample is a vector

(Element-wise nonlinearity)

Local gradient of the function involved:

$$\frac{\partial}{\partial p} F(G(p, H(x))) = DF(G(p, H(x)))) \frac{\partial}{\partial p} G(p, H(x)))$$

Gradients propagated by a backward pass recursively

Forward propagated function values

Backpropagation algorithm

Rummelhart et al, 1986

Sketch of Deep Artificial Neural Network Training

- Sample batch B_i of training examples
- Maintain network parameters

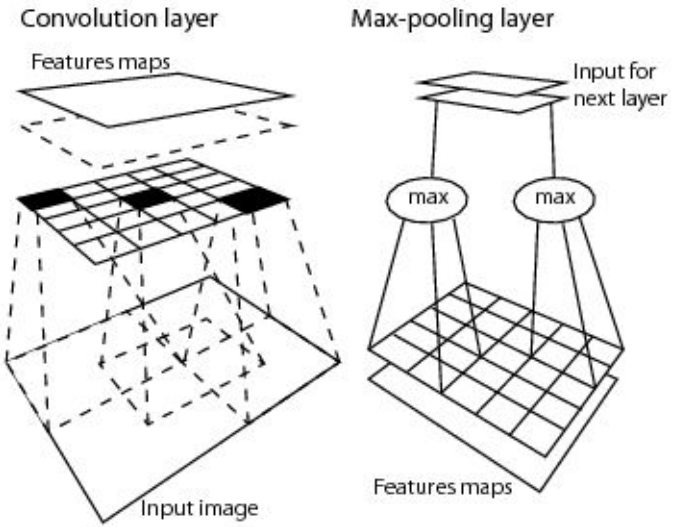
$$M = (W_1, b_1, \ldots, W_n, b_n)$$

- Compute network output $N(v)$ for each training example v
- Compute $loss(N(v))$ of each of the predictions.
- Use backpropagation to compute the gradients g with respect to the model parameters.
- Update M by subtracting αg.

Real Life Deep Network Training

- Data collection and preprocessing and input encoding
- Choosing a suitable framework that can do automatic differentiation.
- Designing suitable network architecture
- Using more sophisticated optimizers
- Implementation optimization:
  - Hardware acceleration, esp. GPU
  - Distributed training using multiple model replicas
- Choose hyperparameters like learning rate and weights for auxiliary losses.

Convolutional Networks

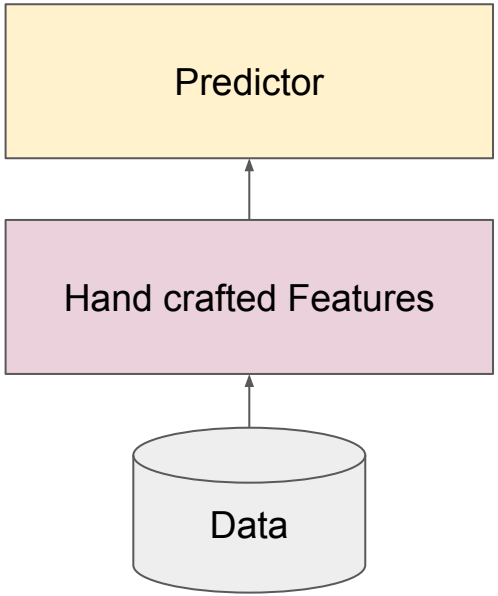


(Image credit: Yann Lecun)

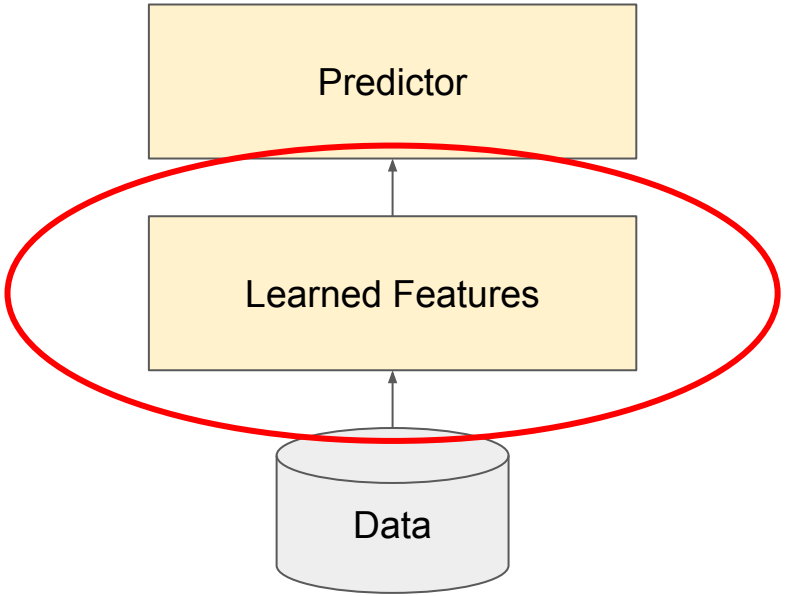
Spatial Parameter-sharing. Neocognitron by [K. Fukushima, 1980].

Convolutional Neural Network, by Yann Lecun et al. (1988).

Deep versus Shallow Learning

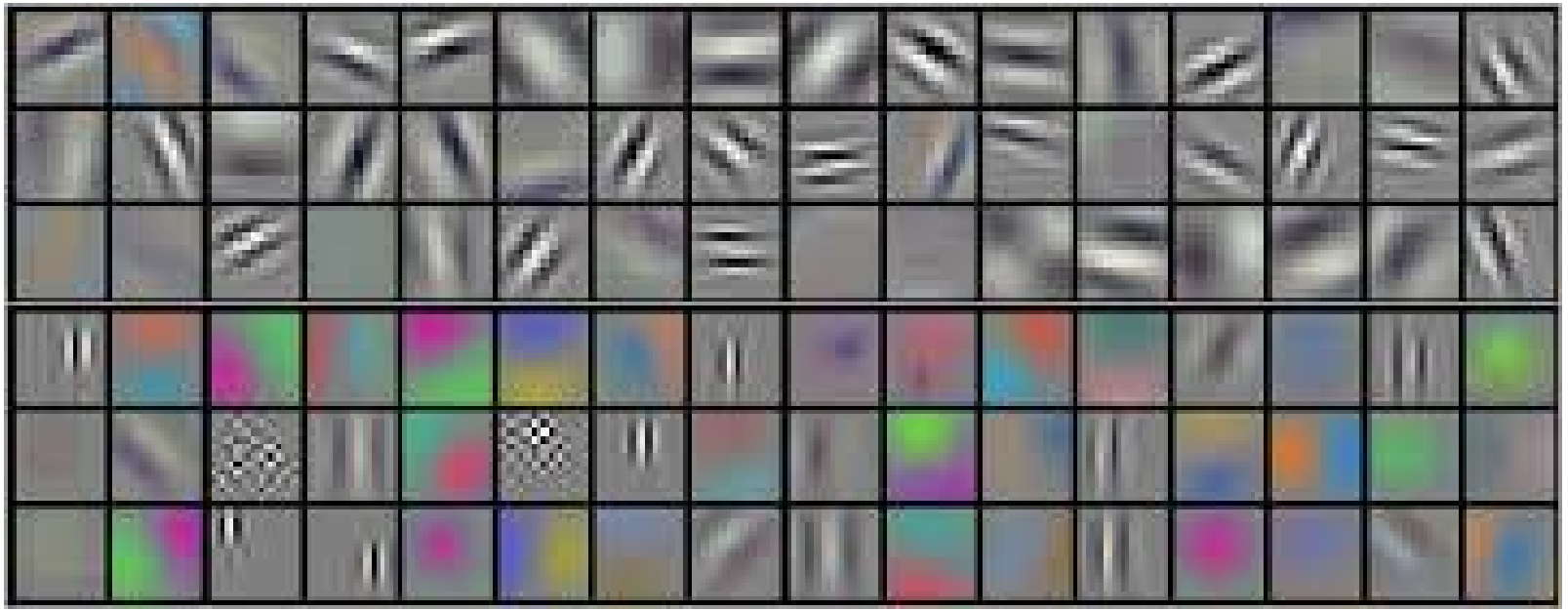


Traditional machine learning



Deep Learning

Low level features learned by vision networks



ImageNet Classification with Deep Convolutional Neural Networks
[Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton 2012]

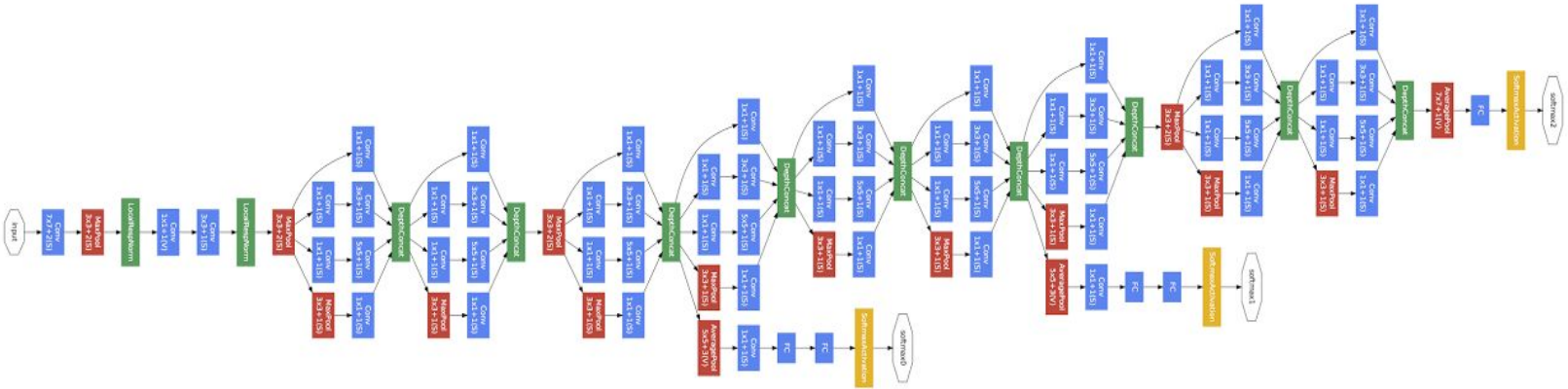
DeepDream visualization of internal feature representation

Starting from white noise image, backpropagate the gradient from a trained network to the image pixel and try to maximize the response of various feature outputs.

[Alexander Mordvintsev, Christopher Olah, Mike Tyka, 2015]

Cambrian Explosion of Deep Vision Research

Zeiler-Fergus Network (ILSVRC winner 2013)



Inception-v1 (GoogLeNet), ILSVRC winner 2014

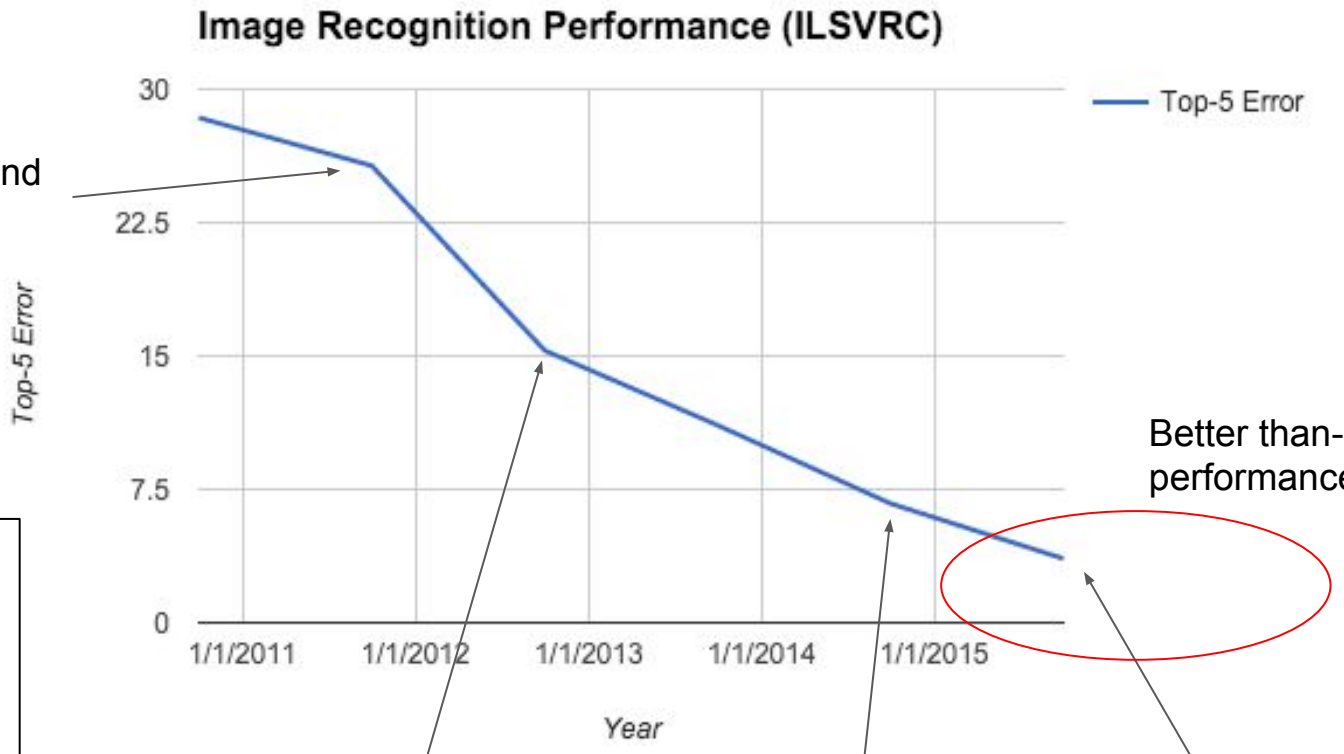


Image Recognition Performance (ILSVRC)
Top-5 Error
30
22.5
15
7.5
0
Top-5 Error
1/1/2011
1/1/2012
1/1/2013
1/1/2014
1/1/2015
Year
Fisher Vectors + Hand crafted features
Better than-human performance
Task: 1000 fine grained classes including the difference between “Eskimo dog” and “Siberian husky”
Convolutional networks
Inception-v1 convolutional network
Residual convolutional network

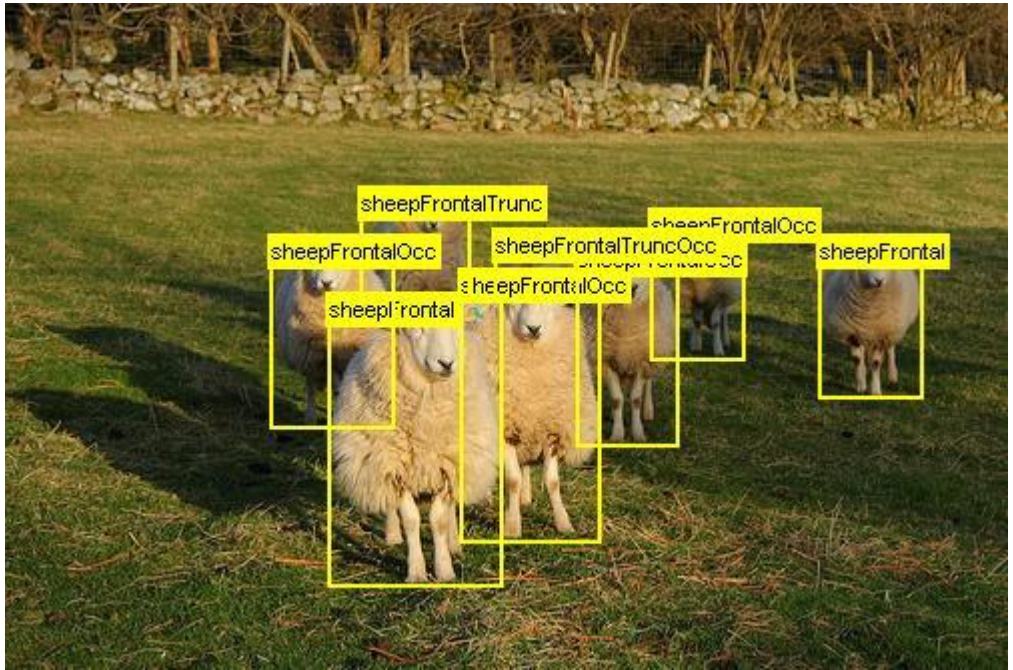
Siberian husky

Eskimo dog

Example images from the ImageNet dataset (**ImageNet Large Scale Visual Recognition Challenge,** IJCV 2015 by Russakovsky et al)

Object Detection

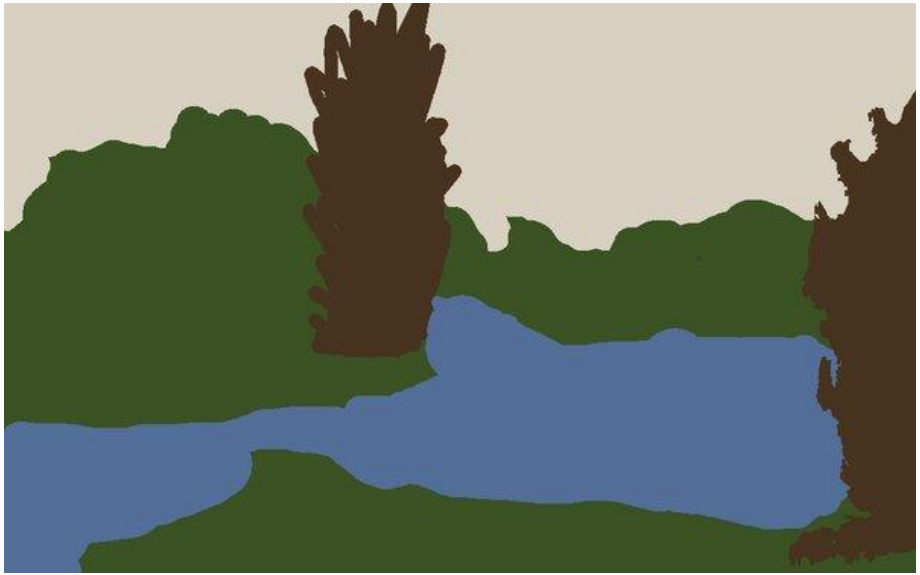
VOC benchmark: detecting objects for 20 different categories (persons, cars, cats, birds, potted plants, bottles, chairs etc.)

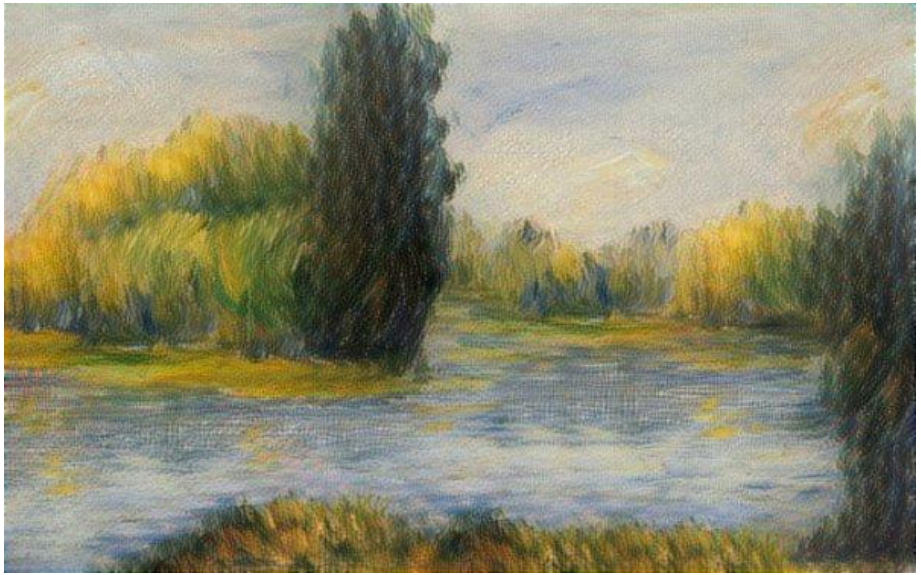


State of the art:

Pre-deep learning in 2013 (Deformable Parts)	Deep-learning model 2015
36% mAP	78% mAP

Stylistic Transfer using Deep Neural Features





Source: Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artwork, nucl.ai Conference 2016 by **Alex J. Champandard** [2016]
http://arxiv.org/pdf/1603.01768v1.pdf

Real Life Applications of Deep Vision Networks

Google Image and Photo Search (Inception-v2)

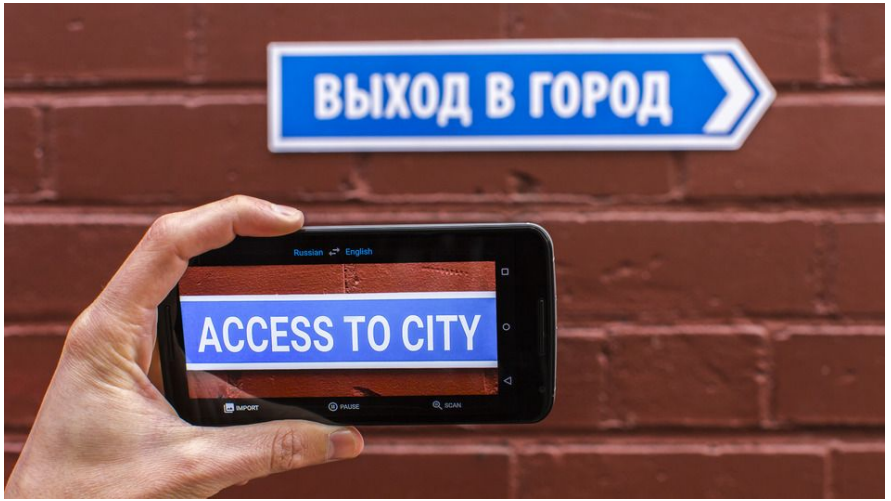
Face detection and tagging in Google photos

PlaNet Identifying the location where image was taken

StreetView privacy protection

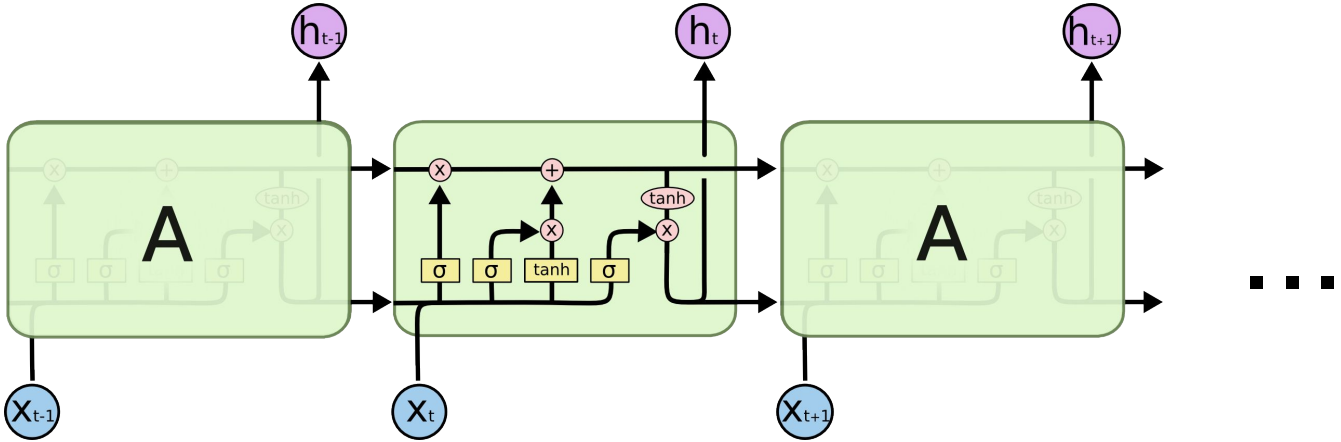
Google Visual Translate

Nvidia’s DriveNet



All of the above applications use variants of the Inception network architecture.

Recurrent Neural Networks



Parameter-sharing over time.

LSTM: Long-short term memory by [Sepp Hochreiter, J□ urgen Schmidhuber, 1997] (Image credit: Cristopher Olah)

http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf

Generative Models of Text

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariconly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$*. Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$*. Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation ??. Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

[Andrej Karpathy 2016]

Some Real life applications of recurrent networks

Voice transcription in phones [Siri, OK Google]
Video Captioning in YouTube
Google Translate
House number transcription from StreetView to Google Maps

Open Source Deep Learning Frameworks

torch

http://torch.ch

- Lua API
- Long history
- GPU backend (via cudnn)
- Most control about dynamic execution
- No support for distributed training

Open Source Deep Learning Frameworks

Theano

http://deeplearning.net/software/theano

- Python API
- University of Montreal project
- Fast GPU backend (via cudnn)
- Less control over dynamic execution than torch
- No support for distributed training

Open Source Deep Learning Frameworks

https://www.tensorflow.org

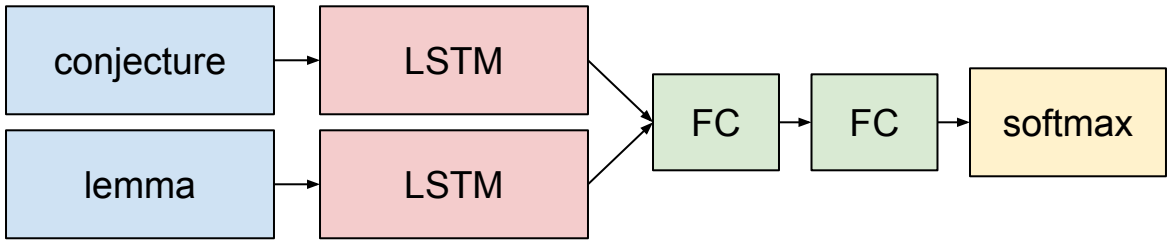
- Python, C++ APIs
- Used and maintained by Google
- Fast GPU backend (via cudnn)
- Less control over dynamic execution than torch
- Support for distributed training now in open source

Deep learning for lemma selection

- Collaboration between
  - Josef Urban’s group
  - Google Research
- Input from the Mizar corpus:
  - Set of known lemmas
  - Proposition to prove
- Pick small subset of lemmas to give to E Prover

Deep learning for lemma selection

- Simplified goal:
  - Rank lemmas by usefulness for a given conjecture
- Embed lemma into $\mathbb{R}^n$ using an LSTM
- Embed conjecture into $\mathbb{R}^n$ using a different LSTM
- Combine embeddings to estimate usefulness



Thank you!